UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,785	01/04/2001	Pierre-Alain Darlet	11283/35	3238

30636 7590 02/01/2008

FAY KAPLUN & MARCIN, LLP
150 BROADWAY, SUITE 702
NEW YORK, NY 10038

EXAMINER	
KISS, ERIC B	
ART UNIT	**PAPER NUMBER**
2192	
MAIL DATE	**DELIVERY MODE**
02/01/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/754,785	DARLET, PIERRE-ALAIN	
	Examiner	Art Unit	
	Eric B. Kiss	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 November 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-60* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-60* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. The reply filed November 1, 2007, has been received and entered. Claims 1-60 are pending.

Response to Amendment

2. Applicant's amendments to the claims appropriately addresses the rejection of claims 1-15 and 40-60 under 35 U.S.C. § 112, first paragraph, as containing new matter. Accordingly, this rejection is withdrawn.

3. Applicant's amendments to the claims appropriately addresses the rejection of claims 1-15, 40, 41, and 43-56 under 35 U.S.C. § 101, and accordingly, this rejection is withdrawn.

Response to Arguments

4. Applicant's arguments filed November 1, 2007, have been fully considered but they are not persuasive.

Levine discloses locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

Levine discloses resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers; see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

Regarding the limitation, "without storing the entire software module in local memory while the symbol reference is resolved," the Board of Patent Appeals and Interferences has already held that Levine anticipates such.

> Here, we have found *supra* that Levine inherently discloses the claimed *target memory space*. We further find that Levine inherently discloses the claimed *local memory* as required for execution of a UNIX® linker and associated *lorder*, *tsort*, and *ar* utilities. Thus, we find that a computer having a "64-bit architecture" (i.e., a *structure* including a *target memory space* and *local memory*) is clearly *capable of* performing the recited negative functional limitation of resolving the at least one symbol reference *without storing the entire software module in local memory while the symbol reference is resolved*" (Claim 16, emphasis added). Accordingly, we find that Levine anticipates independent claim 16 because the absence of a disclosure relating to function does not defeat a finding of anticipation if all the claimed structural limitations are found in the reference.

(Decision on Appeal, May 23, 2007, pp. 9-11 (citing In re Schreiber, 128 F.3d 1473, 1479, 44 USPQ2d 1429, 1433 (Fed. Cir. 1997)).)

Levine discloses instructions operable to reorder components of the software module into a predetermined order (i.e., an order without backward references; see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

Levine discloses the components including at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

Claim Objections

5. Applicant is advised that should claims 1 and 36 be found allowable, claims 38 and 39 will be objected to under 37 CFR 1.75 as being a substantial duplicate thereof. When two claims in an application are duplicates or else are so close in content that they both cover the same thing, despite a slight difference in wording, it is proper after allowing one claim to object to the other as being a substantial duplicate of the allowed claim. See MPEP § 706.03(k).

Claim Rejections - 35 USC § 102

6. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

7. Claims 1-41 and 43-60 are rejected under 35 U.S.C. 102(b) as being anticipated by John Levine, "Linkers and Loaders, chapter 6," June 1999 [online] accessed 08/15/2005, Retrieved from Internet <URL: http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter *Levine*).

As per claim 1, *Levine* discloses receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering components of the software module into a predetermined order to remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

As per claim 2, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 3 and 4, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps

(see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 5-8, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 9, *Levine* discloses a reorder module configured to receive a software module including references to locations within the software module, at least some of the references being backward references, the reorder module configured to reorder components of the software module into a predetermined order and remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), the components including at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header,

followed by alternating file headers and object files)). The use of a processor and memory is inherent in realizing the functionality of *Levine*.

As per claims 10, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 11 and 12, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 13-15, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be

able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 16, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers) without storing the entire software module in local memory while the symbol reference is resolved (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references; see also pp. 2 and 4 (describing a computer structure having a "64-bit architecture" that runs a version of the UNIX® operating system); Decision on Appeal (05/23/2007) at p. 10).

As per claims 17-22, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 23, *Levine* discloses a linker configured to sequentially receive a software module having at least one symbol reference, the linker configured to locate a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)), the linker configured to resolve the symbol reference using the section header table (i.e., using the archive header and file headers), the linker configured to store less than the entire software module in local memory during the resolution of the at least one symbol reference (see "Creating libraries" on pp. 5-6, and in particular, the

discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 24-27, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5; see further, "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 28 and 29, *Levine* further discloses a system symbol table including a field indicative of a defining software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 30 and 31, *Levine* further discloses a software module list (see "Library formats" on pp. 1-5).

As per claims 32-34, *Levine* further discloses storing link status in a symbol table (see "Library formats" on pp. 1-5).

As per claim 35, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 36, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers) without storing the entire software module in local memory at one time (see "Creating

libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 37, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 38, *Levine* disclose receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering the components of the software module into a predetermined order to remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 39, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file

headers) without storing the entire software module in local memory at one time (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claims 40 and 41, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 43-46, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claims 47-54, *Levine* further discloses the reference pointing to/into a section or module before and after reordering (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claim 55, *Levine* discloses receiving a software module, the software module including components arranged in a first order, a first one of the components including a reference to a location in a second one of the components, the second one of the components preceding the first one of the components in the first order; and arranging the components into a predetermined second order so that the second one of the components is subsequent to the first one of the components in the second order (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper

dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

As per claims 56 and 57, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claim 58-60, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

Claim Rejections - 35 USC § 103

8. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

9. Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over *Levine* as applied to claim 1 above, and further in view of U.S. Patent No. 6,185,733 to Breslau et al.

As per claim 42, *Levine* discloses such a method but fails to expressly disclose transferring the reordered module to a different computer system and linking the module on the different computer system. However, *Breslau et al.* teaches the use of remote object libraries distributed prior to linking (see, for example, col. 4, lines 11-20). Therefore, it would have been obvious to one of ordinary skill in the computer art at the time the invention was made to such use of a different computer for linking. One would be motivated to do so, for example, to facilitate distributed software development efforts or reduce the physical storage requirements for object files (see, for example, col. 2, lines 4-25).

Conclusion

10. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

11. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Eric B. Kiss whose telephone number is (571) 272-3699. The Examiner can normally be reached on Tue. - Fri., 7:00 am - 4:30 pm. The Examiner can also be reached on alternate Mondays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Tuan Dam, can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Any inquiry of a general nature should be directed to the TC 2100 Group receptionist: 571-272-2100.

Eric B. Kiss
Primary Patent Examiner
January 29, 2008